Prova di esonero relativa ai capitoli 5 e 6

Esercizio 1. Derivare l'espressione delle formule di quadratura di Newton-Cotes e dire, motivandolo, fino a quale grado è conveniente utilizzarle.

Esercizio 2. Scrivere l'espressione della formula composita dei trapezi e della formula composita di Simpson.

Esercizio 3. Scrivere una function Matlab che implementi efficientemente la formula dei trapezi adattattiva.

Esercizio 4. Scrivere una function Matlab che implementi efficientemente il metodo delle potenze.

Esercizio 5. Definire uno splitting di una matrice nonsingolare A, definire il corrispondente metodo iterativo, e stabilire quando questo risulti essere convergente.

Esercizio 6. Scrivere una function Matlab che sia una implementazione efficiente *ad hoc* del metodo di Jacobi per risolvere il sistema lineare

$$A_N \boldsymbol{x} = \boldsymbol{b},$$

in cui il vettore $\boldsymbol{b} \in \mathbb{R}^N$, $N \geq 5$, è assegnato,

$$A_N = \begin{pmatrix} 4 & -1 & 0 & 0 & 1 & & \\ -1 & 4 & -1 & \ddots & \ddots & \ddots & \\ 1 & -1 & \ddots & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ & & \ddots & \ddots & \ddots & \ddots & 0 \\ & & & \ddots & \ddots & 4 & -1 \\ & & & & 1 & -1 & 4 \end{pmatrix} \in \mathbb{R}^{N \times N},$$

e gli elementi non esplicitati sono da intendersi nulli.

---

1) Per approssimare ($a \le b$, per semplicità)

$I(f) = \int_a^b f(x)\,dx$ si utilizza

$I_n(f) = \int_a^b p_n(x)\,dx$, dove, essendo

$x_i = a + ih$, $i = 0, \ldots, n$, $h = \frac{b-a}{n}$

$p_n \in \Pi_n$ è t.c. $p_n(x_i) = f_i \equiv f(x_i)$, $i = 0, \ldots, n$.

Pertanto, $p_n(x) = \sum_{i=0}^{n} f_i L_{in}(x)$, dove

$L_{in}(x) = \prod_{\substack{j=0 \\ j \ne i}}^{n} \frac{x - x_j}{x_i - x_j}$ è l'i-esimo polinomio di base di Lagrange.

Si ottiene, pertanto,

$$I_n(f) = \int_a^b p_n(x)\,dx = \sum_{i=0}^{n} f_i \int_a^b L_{in}(x)\,dx$$

$$\underset{x = a + th}{=} h \sum_{i=0}^{n} f_i \underbrace{\int_0^n \prod_{\substack{j=0 \\ j \ne i}}^{n} \frac{t-j}{i-j}\,dt}_{= c_{in}} = \frac{b-a}{n} \sum_{i=0}^{n} c_{in} f_i,$$

che definisce la formula di N.-C. di grado n.

Osserviamo che, ponendo $f(x) \equiv 1$, segue che

$\frac{1}{n} \sum_{i=0}^{n} c_{in} = 1$ e, inoltre, per $n \le 7$

$c_{in} \ge 0$, $\forall i = 0, \ldots, n \Rightarrow \frac{1}{n} \sum_{i=0}^{n} |c_{in}| = \frac{1}{n} \sum_{i=0}^{n} c_{in} = 1.$

Se consideriamo il problema del condizionamento, si ottiene, essendo $\tilde{f}(x)$ una perturbazione di $f(x)$:

$$|I(f) - I(\tilde{f})| = \left| \int_a^b f(x) - \tilde{f}(x)\, dx \right| \leq \int_a^b |f(x) - \tilde{f}(x)|\, dx$$

$$\leq \|f - \tilde{f}\| \cdot (b-a),$$

dove $\|f\| = \max\limits_{a \leq x \leq b} |f(x)|$ (assumiamo $f \in C[a,b]$)

D'altronde, si ottiene:

$$|I_n(f) - I_n(\tilde{f})| = \frac{b-a}{n} \left| \sum_{i=0}^{n} c_{in} (f_i - \tilde{f}_i) \right|$$

$$\leq \frac{b-a}{n} \sum_{i=0}^{n} |c_{in}| |f_i - \tilde{f}_i| \leq$$

$$\leq \frac{b-a}{n} \sum_{i=0}^{n} |c_{in}| \, \|f - \tilde{f}\| \equiv (b-a) \|f - \tilde{f}\|,$$

se $c_{in} \geq 0$, $i = 0, \dots, n$, cosa che avviene per $n \leq 7$. Per $n > 7$, $\frac{1}{n} \sum_{i=0}^{n} |c_{in}| \to \infty$, per $n \to \infty$ e, pertanto, non è raccomandabile l'uso delle formule di Newton-Cotes, per $n > 7$.

2) Se $x_i = a + ih$, $i = 0, \ldots, n$, $h = \frac{b-a}{n}$ e denotando con $f_i = f(x_i)$, $i = 0, \ldots, n$, la formula composita dei trapezi è data da

$$I_{1n}(f) = \frac{b-a}{n}\left(\frac{f_0}{2} + \sum_{i=1}^{n-1} f_i + \frac{f_n}{2}\right).$$

Se n è pari, la formula composita di Simpson è data da:

$$I_{2n}(f) = \frac{b-a}{3n}\left(f_0 + 4\sum_{i=1}^{n/2} f_{2i-1} + 2\sum_{i=1}^{n/2-1} f_{2i} + f_n\right).$$

3)
```
function I2= trape(fun, a,b, tol, fa,fb)
%
% I2= trape(fun,a,b,tol)   calcola l'integrale
%                          di fun(x) da a a b
% con tolleranza tol.
if tol<=0, error('tol non ammissibile'), end
if nargin == 4
   fa= feval(fun,a); fb=feval(fun,b);
end
x1= (a+b)/2; f1 = feval(fun,x1); h=(b-a)/2;
I1= h*(fa+fb); I2 = I1+h*f1; err= abs(I2-I1)/3;
if err>tol, tol=tol/2;
  I2=trape(fun,a,x1,tol,fa,f1)+trape(fun,x1,b,tol,...
                                  f1,fb); end, return
```

4)
```
function [lam,X] = pote(A,tol)
%
% [lam,X] = pote(A,tol);   Metodo delle
%                          potenze per deter-
% minare l'autovalore/autovettore dominante
% di una matrice.

if tol<=0, error('tol errata'); end
[m,n]= size(A);
if m~=, error('matrice non quadrata'); end
rng('default'); % reset random number generator
v= rand(n,1);
itmax = n* ceil(-log10(tol)).
lam = inf;
for i=1:itmax
     x= v/norm(v);
     lam0 = lam;
     v = A*x;
     lam = v'*x;
     err = abs(lam-lam0);
     if err <= tol, break, end
end
if err>tol
     warning('tolleranza non soddisfatto');
end
return
```

5) Sia $A \in \mathbb{R}^{n \times n}$ nonsingolare.
Ogni decomposizione di A nella forma

$$A = M - N, \quad \text{con } M, N \in \mathbb{R}^{n \times n}, \ M \text{ nonsingolare},$$

definisce uno "splitting" di A. Se utilizzato per risolvere il sistema lineare

$$Ax = b, \quad \text{di cui sia } x^* \text{ la soluzione},$$

si ottiene il metodo iterativo

$$(1) \ M x_{k+1} = N x_k + b, \quad k = 0, 1, \ldots, \quad \text{con } x_0 \in \mathbb{R}^n \text{ assegnato}$$

Poiché $(2) \ M x^* = N x^* + b$, (e.g., $x_0 = 0$)
sottraendo membro a membro (2) da (1), e definendo $e_k = x_k - x^*$ l'errore al passo k, si ottiene

$$M e_{k+1} = N e_k \Rightarrow e_{k+1} = M^{-1} N e_k, \ k = 0, 1, \ldots$$

Pertanto, $e_k = (M^{-1}N)^k e_0$, $k = 0, 1, \ldots$ ed il metodo risulterà convergente se e solo se la matrice di iterazione del metodo, $M^{-1}N$, è convergente $\Leftrightarrow \rho(M^{-1}N) < 1$.

6)
```
function y = matrec(x)
  y = 4*x;   y(1:end-1) = y(1:end-1) - x(2:end);
  y(1:end-4) = y(1:end-4) + x(5:end);
  y(2:end) = y(2:end) - x(1:end-1);
  y(3:end) = y(3:end) + x(2:end-2);   return
```

```
function x = jacobi( b, tol, itmax)
%
% x= jacobi (b, [tol, itmax]); risolve
%                               il sistema
% lineare A_N x = b, con A_N ∈ R^(NxN),
% N= length(b), e
%
% A_N = toeplitz([4 -1 1 0 ...]', [4 -1 0 0 1 ...])
% con tolleranza tol e max numero di iterazioni
% itmax. Per default,
%
% tol= 1e-6, itmax = ceil((-log10(tol))*N
n= length(b); if n<5, error('n errato'); end
if nargin <=1
    tol = 1e-6; itmax= ceil(-log10(tol))*n;
elseif nargin<=2
    itmax= ceil(-log10(tol))*n;
end
if tol<=0, error('tol errata'); end
if itmax <=0, error('itmax errato'); end
r = b; tolb = tol*norm(b); flag =1; x= 0*b;
for i=1:itmax
    r = matvec(x) - b;
    if norm(r) <= tolb, flag =0; break, end
    u= r/4; x = x-u;
    r = matvec(x) -b;
end
if flag
  warning('tolleranza non raggiunta')
end
return
```